VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Ville Niemelä

MITTAUSJÄRJESTELMÄN KÄYTTÖÖNOTTO

Tekniikka ja liikenne
2014

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Ville Niemelä
Opinnäytetyön nimi	Mittausjärjestelmän käyttöönotto
Vuosi	2014
Kieli	suomi
Sivumäärä	39
Ohjaaja	Jukka Matila

Opinnäytetyö tehtiin Vaasan ammattikorkeakoululle. Työn tarkoituksena on korjata matalaenergiaohjelma, jonka tavoitteena on saada ilmanvaihto, kosteus- ja lämpötila-anturit tuottamaan dataa tietokanta palvelimelle ja sieltä eteenpäin. Tiedonsiirtona mittalaitteelta kerääjälle Technobothnia-laboratoriossa toimii ICPCON I-7000 sarjan RS232-muunnin.

Työ tehtiin Raspberry Pi-alustalle, jonka käyttöjärjestelmänä toimii Raspbian. Ohjelmisto on kirjoitettu Python-kielellä. Työkaluna tälle ohjelmointikielelle toimi Geany.

Lämpötila-anturin ohjelma saatiin toimimaan oikein, mutta ajan vähyyden ja piirilevyjen toimimattomuuden takia kerrostalossa, ilmanvaihto ja kosteusanturien ohjelmia ei saatu toimimaan.

Avainsanat Raspberry Pi, Python, matalaenergia, debuggaus, Geany

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Ville Niemelä
Title	Utilizing data collection system
Year	2014
Language	Finnish
Pages	39
Name of Supervisor	Jukka Matila

The thesis was done for the Vaasan ammattikorkeakoulu, University of Applied Sciences. The purpose of this thesis was to debug the Matalaenergia software, which is intended to receive data from air ventilation, moisture and temperature sensors, and send the information to the database server. In the Technobothnia-laboratory, there is IPCON I-7000 series RS232-converter which does the data transfer from the indicator to the collector.

The thesis was done with Raspberry Pi platform, whose operating system is Raspbian. The Matalaenergia software is written in Python. The tool used for this language was Geany.

The temperature sensor program is now working properly, but due to lack of time and working circuit boards in the building, the air ventilation and moisture sensor programs are not working at the time.

Keywords Raspberry Pi, Python, matalaenergia, debugging, Geany

SISÄLLYS

LYHENTEET

ARM	Advanced RISC Machines, 32-bittinen mikroprosessoriarkkitehtuuri.
CSS	Cascading Style Sheets, WWW-dokumenteille kehitetty tyyliohjeiden laji
CPU	Central Processing Unit, keskusyksikkö
FTP	File Transfer Protocol, tiedonsiirtomenetelmä
GPU	Graphics Processing Unit, grafiikkaprosessori
HTML	Hypertext Markup Language, avoimesti standardoitu kuvauskieli, jota käytetään sivustojen rakentamiseen.
IDE	Integrated Development Environment, integroitu ohjelmaympäristö
IP	Internet Protocol, TCP/IP-mallin Internet-kerroksen protokolla
PHP	Hypertext PreProcessor, ohjelmointikieli
RS-232	Recommended Standard 232, tietoliikenneportti
SD	Secure Digital, muistikorttityyppi
SSH	Secure Shell, tietoliikenneprotokolla
SQL	Structured Query Language, tietokantojen kyselykieli
VNC	Virtual Network Computing, protokolla tietokoneen graafisen käyttöliittymän etäkäyttöön
VPN	Virtual Private Network, virtuaalinen erillisverkko

KUVIOLUETTELO

1 JOHDANTO

Opinnäytetyön tarkoituksena on korjata matalaenergiaohjelma, jonka tavoitteena on jatkuvasti tarkkailla kosteutta, ilmanvaihtoa ja lämpötilaa kahdessa Vaasan Suvilahdessa sijaitsevassa kerrostalossa. Molemmissa taloissa on erilliset tietokoneet ja niissä on melkein samanlaiset ohjelmat. Ainoat erot on muuntimet, ensimmäisessä talossa on IPCON-7000 sarjan RS232 -muunnin ja toisessa talossa on Moxan Uport 1450 sarjan RS232/RS485 -muunnin. Tämän jälkeen siirtää kyseinen tieto tietokantaan, nämä ohjelmat on sovitettu yhteen internetsivuilla, joista näkee tiedot tietokantaan tallennetuista tiedoista.
Alkuperäinen ohjelma on ohjelmoitu Vaasan ammattikorkeakoulussa.

Työssä on käytetty Python-kieltä Raspberry Pi alustalla. Ohjelmassa käytetyn Pythonin versio on 2.7. Pääosaisesti ohjelma jota on käytetty koodin kirjoittamiseen on ollut Geany.

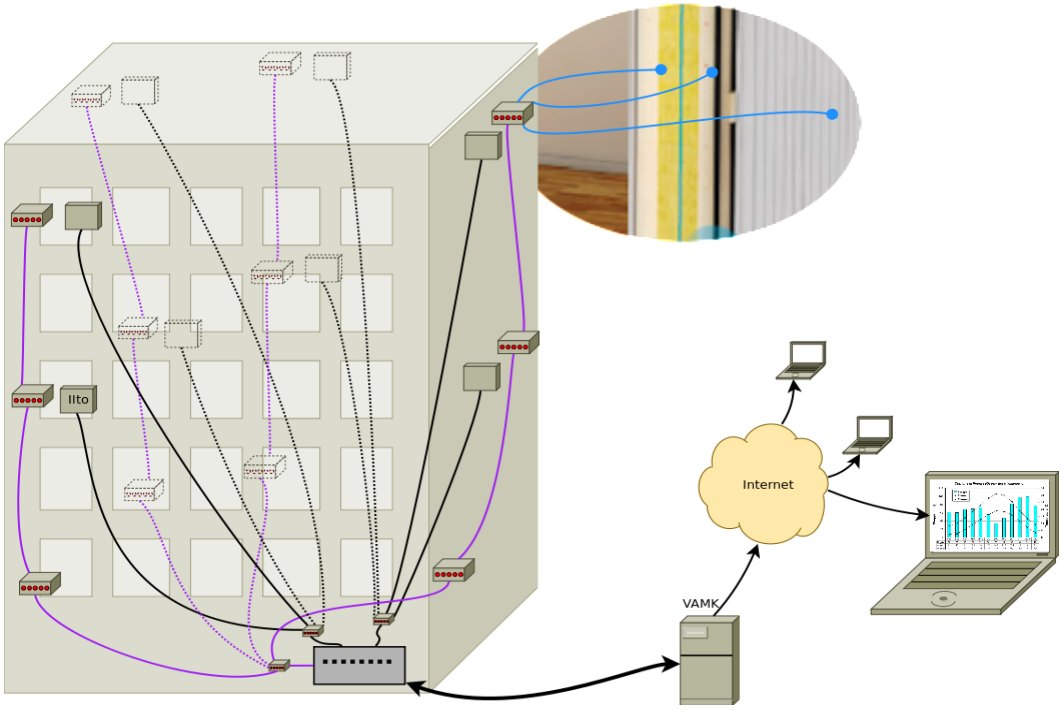
Ennen ohjelman korjauksia täytyi asentaa Raspberry Pi toimintakuntoon ja siihen ohjelmia ja Python moduuleja, joita matalaenergiaohjelma tarvitsi toimiakseen.

Niiden jälkeen alkoikin matalaenergiaohjelman debuggaus ja testaus.

2 LAITTEIDEN TOIMINTA

2.1 Työn kuvaus

Työn tavoitteena on saada Python-kielinen ohjelma toimimaan halutulla tavalla Raspberry Pi -alustalla. Ohjelman tavoitteena on saada ilmanvaihto-, kosteus- ja lämpötila-anturit tuottamaan dataa palvelimelle ja sieltä eteenpäin. **(Kuvio 1.)**



Kuvio 1. Mittausjärjestelmän periaatekuva /7/

Tiedonsiirtona mittalaitteelta kerääjälle Technobothnian laboratoriossa toimii ICPCON I-7000 RS232-muunnin. **(Kuvio 2.)**

Kuvio 2. ICPCON I-7000 RS232-muunnin /8/

2.2 Raspberry Pi

Kuvio 3. Raspberry Pi malli B. /9/

2.2.1 Käyttö

Raspberry Pi:llä ei ole vain yhtä tapaa miten sitä voidaan käyttää. Vaikka tahtoisit vain katsoa videoita ja surffata internetissä, tai haluat hakkeroida ja opiskella, Raspberry Pi on joustava alusta huviin, hyötyyn ja kokeiluun. Koska Raspberry Pi on tietokone niin voit käyttää sitä kuin yhtä. /1/

Kun saat Raspberry Pi:n käyttöön, voit valita käynnistyykö se graafisessa käyttöjärjestelmässä internetselaimen kanssa, jolloin se on paljolti sellainen, mihin me käytämme nykyajan tietokoneita. Voit asentaa siihen paljon ilmaisia ohjelmia, esimerkiksi LibreOfficen (http://www.libreoffice.org/), jota käytetään dokumenteissa ja taulukkolaskennassa. /1/

Koska Raspberry Pi on tarkoitettu koulutustyökaluksi kannustamaan lapsia kokeilemaan ohjelmointia, mukana tulee kääntäjiä monelle erilaiselle ohjelmointikielelle. Aloittelijalle on Scratch, graafinen ohjelmointikieli MIT:stä. Jos olet halukas hyppäämään kirjoittamaan koodia, Python ohjelmointikieli on loistava tapa siihen. Et ole rajoitettu käyttämään vain Scratchia ja Pythonia. Voit

ohjelmoida Raspberry Pi:ssä monella eri ohjelmointikielellä, kuten C, Ruby, Java ja Perl. /1/

Raspberry Pi erottuu normaalista tietokoneesta, ei vain hinnan ja koon takia, vaan myös kyvykkyydessä yhdistyä erinäisiin elektronisiin projekteihin. /1/

2.2.2 Historia

Idea pieneen ja halpaan lapsille suunnattuun tietokoneeseen tuli vuonna 2006, kun Cambridgen yliopiston tietokonelaboratorion kollegat Eben Upton, Alan Mycroft, Jack Lang ja Rob Mullins, alkoivat huolestua siitä, että vuosi vuodelta tietojenkäsittelytieteeseen hakeneiden oppilaiden määrä ja osaamisen taso olivat laskeneet. Vielä vuonna 1990 melkein kaikki hakijat, jotka tulivat haastatteluun, olivat kokeneita ohjelmoijia harrastuksiensa takia. Vuonna 2000 taas tyypillinen hakija oli tehnyt hyvin vähän websuunnittelua. /1/

Jokin oli muuttanut lasten suhtautumisen tietokoneisiin. Joukko ongelmia tiedostettiin: tieto- ja viestintäteknologian alan opetussuunnitelmat sisälsivät lähinnä Wordin ja Excelin käyttöä tai websivujen kirjoittamista. Pelikonsolit ja kotitietokoneet puolestaan syrjäyttivät Amigat, BBC Microt, Spectrum ZX:n ja Commodore 64-koneet, joilla aikaisemmat sukupolvet olivat oppineet ohjelmoimaan. /1/

Pienellä ryhmällä ei ollut paljon mahdollisuuksia käsitellä ongelmia, kuten puutteellista opetussuunnitelmaa tai tiukkaa taloudellista tilannetta. Ryhmä kuitenkin mietti, että he voisivat tehdä jotain sille asialle, kun tietokoneista oli tullut niin kalliita ja vaikeaselkoisia, että ohjelmointikokeilut oli jouduttu kieltämään vanhempien toimesta. Olisi löydettävä samanlainen alusta, kuin vanhoissa kotitietokoneissa, joka pystyttäisiin käynnistämään suoraan ohjelmointiympäristöön. Vuosina 2006 ja 2008 ryhmä suunnitteli useita versioita, joista lopulta syntyi Raspberry Pi. /1/

Vuoteen 2008 mennessä, matkapuhelinten prosessorit oli riittävän tehokkaita ja edullisia tarjotakseen hyvän multimedian. Tämän ominaisuuden takia, ryhmä aavisteli pystyvänsä suunnittelemaan alustan lapsille, joilla ei heti alussa ole

kiinnostusta pelkkään ohjelmointiin. Upton, Mullins, Lang ja Mycroft ryhmittyivät Pete Lomaksen ja David Braben kanssa ja perustivat Raspberry Pi Foundationin. Kolme vuotta myöhemmin, Raspberry Pi Model B **(Kuvio 3.)** meni massatuotantoon, ja vuodessa sitä oli myyty yli miljoona yksikköä. /1/

2.2.3 Käyttöjärjestelmä

Raspbian on ilmainen Rasperry Pi:lle suunniteltu käyttöjärjestelmä, joka perustuu Debianiin. Käyttöjärjestelmässä on tietty määrä tavallisia ohjelmia ja työkaluja, jotka laittavat Raspberry Pi:n toimimaan. Raspbian tarjoaa muutakin, kuin pelkän käyttöjärjestelmän. Siinä tulee mukana yli 35000 esikäännettyä ohjelmapakettia hienossa formaatissa, jotka on helposti asennettavissa Raspberry Pi:lle /2/

Suositeltu käyttöjärjestelmä on Raspbian, joka on erityisesti suunniteltu Raspberry Pi:lle. Mahdollista on myös käyttää muita ARM Linux-jakeluita, kuten Pidora, RaspBMC, OpenELEC, RISC OS ja Arch.

2.2.4 Alkuvalmistelut ja tekniikka

SD-muistikortin alustaminen

Monet jälleenmyyjät myyvät SD-muistikortteja, joissa on käyttöjärjestelmä jo valmiiksi asennettuna. Joillekin ihmisille tämä on paras vaihtoehto alkuun. Vaikka se ei olisi uusin julkaisu, sen voi helposti päivittää uusimpaan internetin välityksellä. /3, 10-11/

Raspbianista on myös verkkoasennuspaketti (http://www.raspbian.org/RaspbianInstaller). Käyttääksesi tätä työkalua, sinun pitää laittaa asennustiedostot SD-muistikortille (alustettu FAT32 muotoon, joka on tyypillinen näille korteille) ja sen jälkeen käynnistää Raspberry Pi SD-muistikortti sisällä. Sinun pitää olla yhteydessä internetiin, jotta tämä toimii. /3, 10-11/

Ensiksi sinun täytyy ladata Raspbian raspberrypi.org sivuilta (http://www.raspberrypi.org/downloads). Käyttöjärjestelmä on jaettu levykuvana,

joka on bitti bitiltä esitys siitä kuinka, datan pitäisi kirjoittaa SD-muistikortille. /3, 10-11/

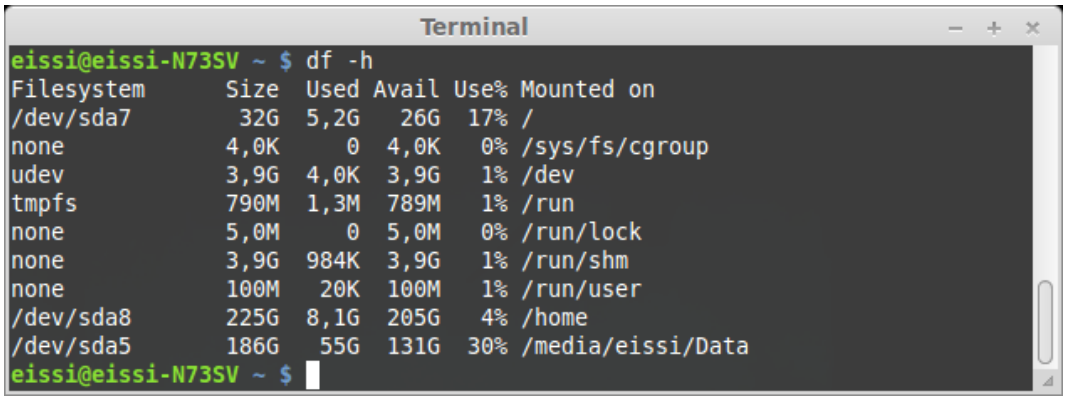
Huomaa, että et voi vain viedä levykuvaa SD-muistikortille. Sinun pitää tehdä bitti bitiltä kopio levykuvasta. Tarvitset kortinkirjoittajan ja levykuvatyökalun, mikä tahansa halpa kortinkirjoittaja käy. Ohjeet vaihtelevat sen mukaan mitä käyttöjärjestelmää käytät. Pura levykuvan tiedosto (sinulla pitäisi olla sen jälkeen .img tiedosto). /3, 10-11/

SD-muistikortin kirjoittaminen Windowsilla

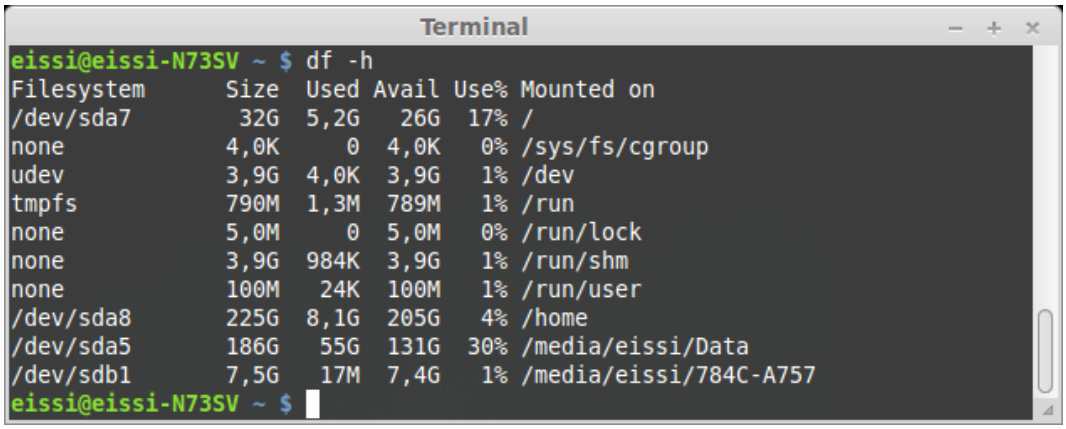
Lataa Win32DiskImager -ohjelma (http://sourceforge.net/projects/win32diskimager/). Laita SD-muistikortti lukijaan ja huomioi aseman kirjain, joka ilmestyy Windows Exploreriin. Avaa Win32DiskImager ja valitse Raspbian levykuva. Valitse SD-korttiaseman kirjain, sen jälkeen paina 'Write'. Jos Win32DiskImagerilla on ongelmia kirjoittaa kortille, voit alustaa sen Windows Explorerissa. Poista SD-muistikortti ja laita se Raspberry Pi:hin kiinni. /3, 146/

SD-muistikortin kirjoittaminen Linuxilla

Avaa Shell-ikkuna ilman SD-muistikorttia lukijassa. Kirjoita df –h nähdäksesi, mitkä asemat ovat käytössä. Laita SD-muistikortti asemaan ja kirjoita df –h uudestaan. **(Kuvio 4. ja 5.)** Katso listasta käytössä olevia asemia ja määrittele, mikä on SD-muistikortti vertaamalla molempia listoja. Etsi laitteen nimi, jonka pitäisi olla tyyliin /dev/sdd1. Riippuen tietokoneesi konfiguraatiosta, tämä nimi voi vaihdella. Ota kortin laitteen nimi ylös. Kirjoittaaksesi korttiin, sinun pitää irrottaa se ensiksi. Irrota se kirjoittamalla unmount /dev/sdd1 (käyttämällä siis sitä laitteen nimeä, jonka sait äskeisestä osioista, ei pelkästään /dev/sdd1). Jos korttia ei voi irrottaa, varmista ettei se ole käytössä missään Shell-ikkunassa. Seuraavaksi sinun pitää keksiä kortin käsittelemätön laitteen nimi, joka on laitteen nimi ilman osio numeroa. Esimerkiksi, jos laitteesi nimi oli /dev/sdd1, niin käsittelemättömän laitteen nimi on /dev/sdd. /3, 147/



Kuvio 4. df –h ilman SD-muistikorttia



Kuvio 5. df –h SD-muistikortin kanssa

On tärkeää, että saat käsittelemättömän laitteen nimen oikein. Voit vahingossa päällekirjoittaa kovalevylle ja menettää dataa, jos aloitat siihen kirjoittamisen etkä SD-muistikortille. Käytä df -komentoa uudestaan tupla tarkistamiseen, ennen kuin jatkat. Varmista, että ladattu levykuva on purettu ja sijoitettu koti kansioon. Tulet käyttämään Unixin omaa työkalua ’dd’ levykuvan kopioimiseen SD-muistikortille. Alempana on komento kyseiselle toiminnalle. Vaihda vain nimi, levykuvan nimi siihen, mitä latasit ja vaihda /dev/rdisk3 käsittelemättömään SD-muistikortin nimeen. /3, 147/

sudo dd bs=1M if=~2012-09-18-wheezy-raspbian.img of=/dev/sdd

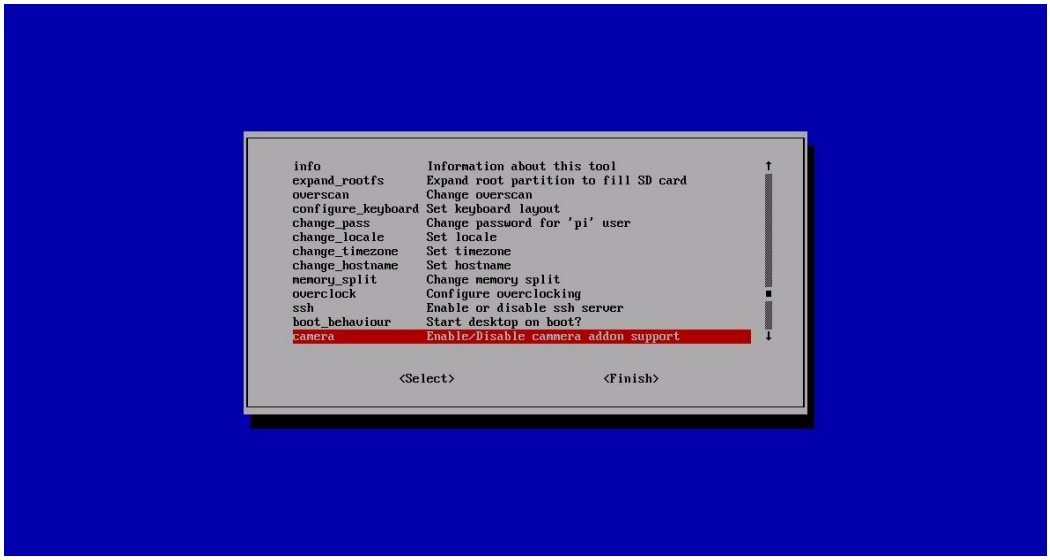
Tämä komento kertoo dd:tä ajamaan pääkäyttäjänä ja kopioimaan syötetyn tiedoston (if) tuotettavaksi tiedostoksi (of). Tämä toiminta tulee kestämään

muutaman minuutin, ennen kuin se on kopioinut koko levykuvan. Valitettavasti dd ei tarjoa mitään visuaalista tulosta, joten sinun täytyy vain odottaa. Kun se on valmis, näyttää se jotain statistiikkaa. Kortin irrottamisen pitäisi olla mahdollista nyt, mutta varmistaaksesi, että se on turvallista, aja sudo sync-komento, joka ajaa ulos järjestelmän kirjoituspuskurit. Irrota SD-muistikortti ja laita se Raspberry Pi:hin kiinni. /3, 147/

Käynnistäminen:

Laita SD-muistikortti korttipaikkaan, jonka jälkeen laita USB-näppäimistö ja hiiri kiinni. Laita HDMI-näyttö tai TV kiinni. Käynnistä näyttö. Laita pistoke pistorasiaan. Yleisesti yritä varmistaa, että kaikki on kiinni ennen käynnistämistä.

Raspberry Pi:n konfigurointi **(Kuvio 6.)**



Kuvio 6. Raspberry Pi:n konfiguraatiosivu /10/

Expand Filesystem: Sinun tulisi aina valita tämä vaihtoehto. Tämä suurentaa tiedostojärjestelmää niin, että voit käyttää koko SD-muistikortin.

Change User Password: On hyvä idea vaihtaa oletussalasana raspberrystä johonkin vahvempaan.

Enable Boot to Desktop: Tämä asetus antaa käynnistää suoraan graafiseen työpöytään ja on oletuksena päällä. Jos valitset sen pois päältä, saat komentolinjan, kun käynnistät Raspberry Pi:n ja sinun tulee kirjautua sisään ja käynnistää graafinen liityntä näin:

raspberry login: pi
Password: raspberry
pi@raspberrypi ~ $ startx

Internationalisation Options -asetuksessa voit vaihtaa aikavyöhykkeen ja näppäimistön asetelman.

Overclock -asetus antaa sinulle mahdollisuuden laittaa prosessori pyörimään nopeammin kuin 700MHz. Jätä se ensikäynnistyksen ajaksi oletusasetukselle tai kokeile medium-asetusta tai modest-asetusta. Voit myöhemmin haluta vaihtaa tämän (Turbo-moodi voi pyöriä 1000MHz nopeudella).

Advanced Options: Overscan -asetuksen voit jättää pois päältä. Jos sinulla on teräväpiirtonäyttö, niin voit huomata, että teksti menee reunojen yli. Korjataksesi tämän, laita overscan päälle ja vaihda arvoja niin, että kuva sopii näyttöön. Arvot osoittavat sen määrän, mitä näytön ohjelma voi korjata. Käytä positiivisia arvoja, jos kuva menee näytön yli ja negatiivisia arvoja, jos näytön reunoilla on mustat rajat.

Memory Split -asetus antaa sinulle mahdollisuuden valita kuinka paljon muistia CPU ja GPU käyttää. Jätä tämä oletusjaolle.

SSH -asetus käynnistää SSH-serverin, jonka avulla pääset Raspberry Pi:hin sisälle tietoverkon yli. Tämä on hyvin kätevää, joten se kannattaa laittaa päälle.

Update –asetuksella voit päivittää config -työkalun jos olet yhteydessä internetiin.

Sammuttaminen:

Raspberry Pi:ssä ei ole erinäistä käynnistysnappulaa. Kunnollinen tapa sammuttaa Raspberry Pi, on käynnistysvalikossa olevasta ’Logout’ -menusta graafisessa työpöydässä. Valitse Shutdown pysäyttääksesi laitteen.

Voit myös pysäyttää laitteen komentolinjan kautta kirjoittamalla:

sudo shutdown –h now

Muista tehdä kunnollinen sammutus (etkä vain ota pistoketta pois). Joissain tapauksissa voit vahingossa korruptoida SD-muistikortin, jos sammutat laitteen ilman pysäyttämistä.

Uusien ohjelmien asentaminen

Yksi alue missä Linux lyö muut käyttöjärjestelmät, on sen ohjelmanhallintapaketti. Hallintapaketti hoitaa ohjelman latauksen ja asentamisen, se automaattisesti hoitaa myös riippuvaisten asioiden latauksen.

Raspbianissa tulee itsessään aika minimaalinen määrä ohjelmia, joten joudut pian lataamaan ja asentamaan uusia ohjelmia.

Apt-get käskyä –install argumentin kanssa käytetään ohjelman lataamisessa. Apt-get lataa kaikki tarvittavien ohjelmien paketit, joten sinun ei itse tarvitse lähteä etsimään niitä. Ohjelma täyty asentaa ylläpitäjän oikeuksilla, joten käytä aina sudo-komentoa.

pi@raspberrypi ~ $ sudo apt-get install emacs (Tämä komento asentaa Emacs tekstieditorin.)

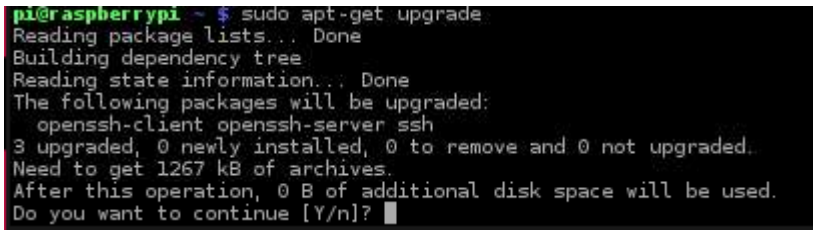
Raspberry Pi:n päivittäminen

Ennen kuin alat asentamaan uusia ohjelmia, on hyvä varmistaa, että kaikki on päivitetty. Varmista, että sinulla on Internet-yhteys ja aja seuraavat käsky:

sudo apt-get update

Jonka jälkeen kirjoitat seuraavan komennon:

sudo apt-get upgrade



Kuvio 7. Pakettien päivitys

Nyt on myös hyvä aika varmistaa, että koko Linux-jakelu on ajan tasalla, komennolla:

sudo apt-get dist-upgrade

Voit ajaa nämä käskyt aina silloin tällöin varmistaaksesi, että sinulla on päivitetyt ohjelmat. /3, 10-32/

2.3 Ohjelmointi Raspberry Pi:llä

2.3.1 Python kielenä

Python on korkea-arvoinen, avoin lähdekoodinen ohjelmointikieli, jota voidaan käyttää monella eri tavalla ohjelmoitaessa tehtäviä. Pythonin on luonut Gudio Van Rossum 1990-luvun alussa, ja sen seuranta on kasvanut tasaisesti ja kiinnostus on noussut selvästi muutamina viime vuosina. Se on nimetty Monty Python’s Flying Circus komedia ohjelman mukaan. /4/

Yksi hyvä piirre Pythonissa on sen saatavuus joka alustalle. Pythonia pystyy ajamaan helposti Microsoft Windowsissa, Macintoshissa ja kaikissa Linux-jakeluissa. Tämä tekee siitä helposti siirrettävän, kun jokainen ohjelma, joka on kirjoitettu tietylle alustalle, voi käyttää sitä toisessa alustassa. /4/

Python on hyvä ensimmäiseksi ohjelmointikieleksi, sillä se on selkeä ja helppo asentaa. Python on tulkinnallinen kieli ja sillä voi kirjoittaa ohjelman, skriptin tai suorittaa sitä suoraan eikä vain kääntää sitä konekieleksi. Tulkinnallisilla kielillä on hieman nopeampi ohjelmoida ja ne tarjoavat muutamia etuja. Esimerkiksi

Pythonilla sinun ei tarvitse selvästi kertoa onko muuttuja numero, lista vai merkkijono. Tulkki tajuaa datatyypin kun ajat koodin. /3/

2.3.2 Ohjelmointi Pythonilla

Ensimmäiseksi sinun täytyy asentaa Python tietokoneellesi. Voit ladata sen Pythonin kotisivuilta (http://www.python.org/download). Linuxin, BSD:n ja unixin käyttäjillä se on luultavasti jo valmiina asennettuna. Varmistuaksesi siitä, kirjoita ’python’ komentolinjaan. Jos näet jotain seuraavassa osassa, olet valmis. Jos sinulla ei ole Pythonia asennettuna, käytä käyttöjärjestelmän ohjelmanhallintapakettia asentaaksesi sen.

Linuxilla voit käyttää joko IDLEa tai jotain muuta ohjelmointiympäristöä, kuten Geanya.

Windows-käyttäjillä voi olla esiasennettuna Python. Tarkistaaksesi onko se asennettuna, avaa komentolinja (Windows näppäin + R ja kirjoita cmd) jonka jälkeen kirjoitat siihen ikkunaan ’python’. Jos siihen tulee teksti ”Bad command or file name”, sinun täytyy ladata Windowsille oma asennuspaketti.

Asennuksen jälkeen sinun tulee laittaa Pythonille PATH -ympäristömuuttuja Windowsiin. Ohjauspaneelista valitset ’Järjestelmä’ ja sieltä valitset ’Järjestelmän lisäasetukset’. Sieltä valitset ’Ympäristömuuttujat’ ja uudesta ikkunasta etsit Path -muuttujan. Editoi sitä niin, että kirjoitat sinne esimerkiksi Python 2.7 versiolle ;c:\python27;

Tämän jälkeen voit joko käyttää IDLEa, Notepad++:a tai jotain muuta ohjelmointiympäristöä. Ajaaksesi Python ohjelmasi, voit käyttää esimerkiksi komentolinjaa:

```
Python 2.7.6 (default, Nov 10 2013, 19:24:24) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print "Hello, World!"
Hello, World!
>>>
```

Kuvio 8. Python Hello, World! esimerkki.

2.3.3 Geany

Geany on pieni, kevyt ja ilmainen ohjelmointiympäristö. Se on kehitetty tarjotakseen pienen ja nopean ohjelmointiympäristön, jolla on vain muutamia riippuvaisuuksia muista paketeista. Geany **(Kuvio 9.)** toimii muun muassa Linuxilla, OpenBSD:llä, MacOS X:llä ja Windowsilla. Sen pitäisi toimia jokaisella alustalla, joka tukee GTK -kirjastoja. /5/

Tuettuja ohjelmointi kieliä on esimerkiksi C, C++, C#, Java, Javascript, PHP, HTML, CSS, Python, Perl, Ruby ja Pascal.

Kuvio 9. Geany logo. /11/

3 Käytäntö

3.1 Perustiedot

Matalaenergiaohjelma tuottaa ilmanvaihto, kosteus- ja lämpötila-antureista dataa ja kääntää tämän datan ymmärrettävään muotoon. Tämän jälkeen se lähetetään tietokantaan, josta sen voi lukea suoraan sieltä, tai esimerkiksi erikseen tehdystä sivustosta.

Pythonilla tekemäni SQL-ohjelma ottaa rivin kerrallaan tekstitiedostosta muokaten sitä ja lähettää sen jälkeen tiedot tietokantaan.

3.2 Toteutus

Jotta matalaenergiaohjelma saataisiin toimimaan Raspberry Pi:llä, tarvittiin seuraavanlaiset Python kirjastot: PyMySQLdb, Pexpect, pySerial.

PyMySQLdb on käyttöliittymä MySQL -tietokanta palvelimelle. Jotta ohjelma saa tietokantaan yhteydet, ja pystyy lähettämään sinne tietoa. Python MySQL tietokanta paketin voi asentaa komennolla:

sudo apt-get install mysql-server python-mysqldb

Pexpect on Python moduuli jota voidaan käyttää automatisoimaan interaktiivisia ohjelmia, kuten SSH, FTP, passwd, telnet jne. Jos ohjelma ei saa yhteyttä tietokantaan suoraan, ohjelma voi ottaa yhteyden SSH:n avulla. Pexpect asennetaan seuraavilla komennoilla:

wget http://pexpect.sourceforge.net/pexpect-2.3.tar.gz
tar xzf pexpect-2.3.tar.gz
cd pexpect-2.3
sudo python ./setup.py install

Python pip on työkalu Python pakettien asentamiseen ja hallitsemiseen. Asensimme Python pipin komennolla:

sudo apt-get install python-pip

Tämän jälkeen päivitimme Pythonin paketit komennolla:

sudo easy_install –U distribute

PySerial on Python moduuli, joka kiteyttää pääsyn sarjaportille. Asensimme Pythonille pyserialin komennolla:

sudo pip install pySerial

Asensimme VNC –palvelimen Raspberry Pi:lle. Tällöin pystyimme ottamaan siihen etäyhteyden ja saamaan käyttöliittymän. VNC asennetaan komennolla:

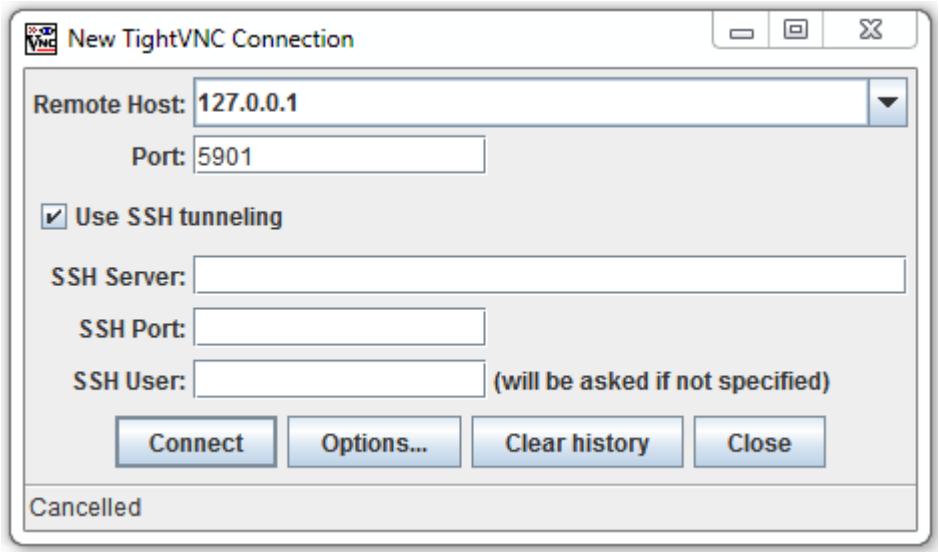
sudo apt-get install tightvncserver

Muokkasimme tightvnc-palvelinta seuraavilla komennoilla siten, että se käynnistyy automaattisesti kun Raspberry Pi käynnistyy.

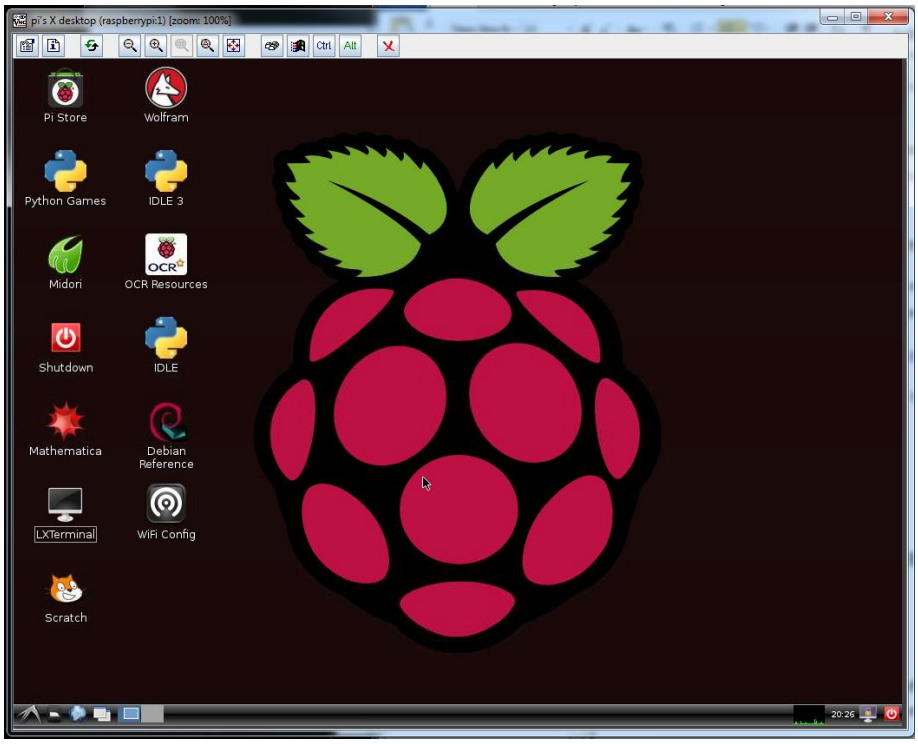
wget http://www.penguintutor.com/otherfiles/tightvncserver-init.txt

sudo mv tightvncserver-init.txt /etc/init.d/tightvncserver

Käytimme TightVNC Java Vieweriä **(Kuvio 10. ja 11.)** Windowsilla saadaksemme graafisen etäyhteyden. Remote Host:in laitetaan Raspberry Pi:n IP-osoite ja porttiin 5901. Jos et käytä SSH –yhteyttä, niin jätä 'Use SSH tunneling' pois päältä. Muuten laita siihen SSH -yhteyden tiedot.



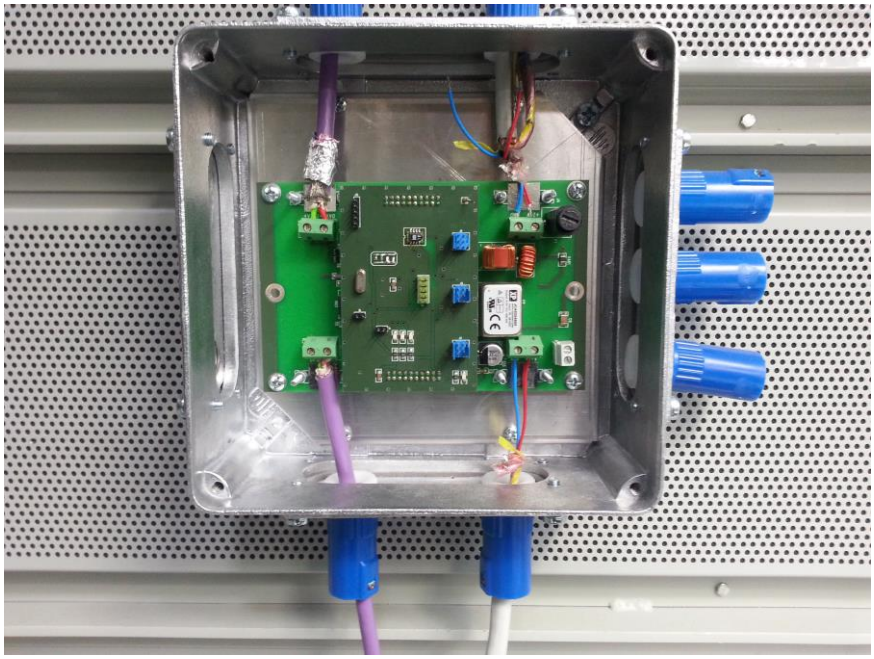
Kuvio 10. TightVNC yhteydenluoja.



Kuvio 11. TightVNC Java Viewer etäkäyttöohjelma.

Kytkettäessä Raspberry Pi:tä Suvilahden kerrostaloon, tarvittiin OpenVPN -ohjelma, jolla saadaan yhteys mysql.cc.puv.fi palvelimille. Laitoimme myös siihen kiinteän IP-osoitteen, jotta siihen saadaan yhteys ilman näyttöä ja näppäimistöä, kun tiedetään IP-osoite valmiiksi.

Technobothnia laboratoriossa kytkin Raspberry Pi:n IPCON I-7000 sarjan RS232 muuntimeen, joka oli kiinni kahdessa mittalaitteessa **(Kuvio 12.)**, jotka toimivat 24-voltin jännitteellä. Nämä mittalaitteet lähettävät kyseltäessä dataa Raspberry Pi:lle, joka sitten siirretään tietokantaan.



Kuvio 12. Lämpötila-anturi

3.2.1 Vianetsintä

Vianetsintä on usein pelkästään eliminointiprosessi, jossa poistetaan mahdolliset ongelmien aiheuttajat. Vianetsintä ohjelmistotuotannossa yleisesti tarkoittaa sitä, että paikallistetaan virhe ja korjataan se.

Seuraavat askeleet toimivat vianetsinnässä:

1. Etsitään miksi ohjelma ei toimi niin kuin pitäisi.
2. Korjataan ongelma.
3. Vältetään hajottamasta mitään muuta.
4. Parannetaan koodin toimivuutta jos mahdollista.
5. Varmista, ettei sama ongelma toistu muualla.

Ei saa unohtaa, että epähuomiossa syntyy usein eniten virheitä, ja ne saattavat olla hyvinkin yksinkertaisia asioita. Joten on hyvä tarkistaa yksinkertaisimmat asiat ensimmäiseksi.

Yksi helpoimmista tavoista etsiä vikoja koodista on laittaa print –komentoja koodiin ja lukea tulostuksista, että mihin kohtaan ohjelma pysähtyy.

On olemassa nykyään paljon erilaisia debugger -ohjelmia, kuten esimerkiksi Pythonille pdb -moduuli. Jonka tuomalla koodiin ja laittamalla toimimaan ’pdb.set_trace()’ –komennolla, se alkaa seurata koodia siitä kohtaa, ja lukee jokaisen rivin kerrallaan kun painat Enter, vian sattuessa ohjelma kertoo siitä. /6/

Tätä ohjelmaa debuggaamalla aloitin toiminta häiriöiden etsinnän terminaali -komennoilla ja print –komennoilla, sen jälkeen luin tulostukset. Niistä päättelemällä löysin ongelmat, joiden takia ohjelma ei toiminut. Joskus se oli suoraan se virhe, jonka tulostus antoi. Toisinaan virhe oli jossain muualla kuin ajetussa koodissa, esimerkiksi toisessa koodissa, jota ajettava ohjelma tarvitsee toimiakseen.

3.2.2 Virheiden korjaaminen

Aloitettiin muokkaamalla globals.py -tiedostoon uudet MySQL-palvelimen tiedot vanhojen tilalle.

Kuviossa 13 esitetään ohjelman ensimmäinen ajokerta. **(Kuvio 13.)** Joka tulostaa ensimmäisen virheen: ’IOError: [Errno 2] No such file or directory’ Joka tarkoittaa sitä, ettei ohjelma pysty avamaan kyseisessä polussa olevaa tiedostoa.

```
$ python Multible_com.py
Traceback (most recent call last):
  File "Multible_com.py", line 17, in <module>
    File_Sensor_start.file_sensor_open()
  File "/home/pi/matala_energia/Project_Python_source_file/Project_Python_source_file/File_Sensor.py",
  line 31, in file_sensor_open
    self.f = open(self.f_name_juma_log, 'a')
IOError: [Errno 2] No such file or directory: '/home/gao/gao_test/data/modbus_logger_juma_kosteus3_2014.log'
```

Kuvio 13. Virheilmoitus 1.

Tässä virheessä se ilmoittaa, että tiedostopolut evät ole koodissa oikein. Koska tämä ohjelma on suoraa otettu vanhasta tietokoneesta, jossa polut olivat eri lailla. Tämän takia pitää laittaa tiedostopolut kuntoon File_Sensor.py tiedostossa. **(Kuvio 14.)**

```
self.f_name_juma_log="/home/gao/gao_test/data/modbus_logger_juma_kosteus%s_%s.log"%(time.today.month,time.today.year)
self.f_name="/home/gao/gao_test/data/mb_log_%s_%s.log"%(time.today.month,time.today.year)
self.f_name_sql_log="/home/gao/gao_test/data/sql_log_%s_%s.log"%(time.today.month,time.today.year)
```

Kuvio 14. Polkujen muokkaus 1.

Tiedostopolkujen tilalle laitoin /home/pi/me_test/data/

Tämän jälkeen ajettiin Multible_com.py:n uudestaan, josta tuli seuraava virhe: **(Kuvio 15.)** ’IOError: [Errno 2] No such file or directory’. Eli tiedostopolku on virheellinen.

```
* Read counter from local file error
* Read counter from local file error
 * SSH error and can not read from local file
Traceback (most recent call last):
  File "Multible_com.py", line 19, in <module>
    conne = db_start.connect()
  File "/home/pi/matala_energia/Project_Python_source_file/Project_Python_source_file/MySQL_Sensor.py", line 72, in connect
self.store_date()
  File "/home/pi/matala_energia/Project_Python_source_file/Project_Python_source_file/MySQL_Sensor.py", line 112, in store_date
    self.f_time = open(self.f_name_time, 'w')
IOError: [Errno 2] No such file or directory: '/home/gao/gao_test/data/time.log'
```

Kuvio 15. Virheilmoitus 2.

Samanlainen ongelma kuin aikaisemminkin, pitää muokata polut uudestaan MySQL_Sensor.py tiedostoon. **(Kuvio 16.)**

```
self.f_name_counter="/home/gao/gao_test/data/counter.log"
self.f_name_time="/home/gao/gao_test/data/time.log"
```

Kuvio 16. Polkujen muokkaus 2.

Laitettiin sama polku kuin aikaisemminkin, eli /home/pi/me_test/data/

Seuraavalla ajokerralla ohjelma näytti seuraavanlaisen virheen: **(Kuvio 17.)** ’Serial was not started, abort!’ Tämä tarkoittaa sitä, että ohjelma ei pysty aukaisemaan porttia.

```
python Multible_com.py
I give password Input password
(counter is going to be updated)
previous counter is 34. if successed it is going to be 35
 1, Date stored
 2, Database connected
 3, Managed to function.
 * Serial (/dev/ttyUSB0) can be started
Serial  was not started, abort !
update moisture_counter set counter='35';
updated!
```

Kuvio 17. Sarjaportin virhe.

Tästä johtuen kirjoitin ohjelman koodiin komennon, joka käskee ohjelmaa printtaamaan portin, jota se yrittää aukaista. **(Kuvio 18. ja 19.)**

```
    def ser_performation(self, time, file, db_start):
        global  information
        for port in range(0, 1):
            for addr in addreses[port]:
                print ports[port]

                file.mb_log.write("\n"+time.time_formatted+" (%d) " % (db_start.counter))
                try:
                    self.ser.port=ports[port]
                except:
                    print "Serial  was not started, abort !"
                    return
                message="%02x%02x%04x%04x" %(addr,3,0,8)
```

Kuvio 18. Portin tulostuskoodi.

Ser_performation katsoo Globals.py koodista portit ja osoitteet, ja lähettää niihin kyselyt. Porttien välimatka on 1, koska laboratoriossa ei ole kuin yksi RS232-muunnin kytkettynä Raspberry Pi:hin kiinni. Se tulostaa portin nimen ja kirjoittaa mb_log tiedostoon ajan ja laskurin numeron. Jos ohjelma ei pysty avaamaan porttia, se tulostaa ”Serial was not started, abort!” ja lopettaa toimintansa.

```
python Multible_com.py
I give password Input password
(counter is going to be updated)
previous counter is 34. if successed it is going to be 35
 1, Date stored
 2, Database connected
 3, Managed to function.
 * Serial (/dev/ttyUSB0) can be started
/dev/ttyMXUSB0
Serial  was not started, abort !
update moisture_counter set counter='35';
updated!
```

Kuvio 19. Portin tulostus.

Huomattiin, että jossain on porttivirhe, minkä takia sarjaportti ei aloita toimintaansa. Löydettiin kyseisen portin Globals.py -tiedostosta ja vaihdoin /dev/ttyMXUSB0 tilalle /dev/ttyUSB0:n. **(Kuvio 20.)** Tämä piti vaihtaa sen takia, koska ohjelma tässä tietokoneessa käyttää IPCON-7000 sarjan RS232-muunninta, eikä Moxan Uport 1450 –sarjan RS232/RS485 muunninta, kuten toinen ohjelma.

```
addreses =[[2,6,10],[3,7,11],[4,8,12],[1,5,9]]

#ports = ['/dev/ttyMXUSB0']
ports = ['/dev/ttyUSB0']

sensor_location=[
                [1,2,0,4],
                [1,2,0,4],
```

Kuvio 20. Globals.py portit.

Seuraavalla ajolla tuli seuraavanlainen virhe: **(Kuvio 21.)** ’Warning: Data truncated for column ’dt’ at row 1’ Joka tarkoittaa sitä, että tietokannan ’dt’ kolumnissa on jotain väärin, se ei pysty tallentamaan niin isoa tietoa sinne.

```
python Multible_com.py
I give password Input password
previous counter is 53. if successed it is going to be 54
 1, Date stored
 2, Database connected
 3, Managed to function.
 * Serial (/dev/ttyUSB0) can be started
:0203000000080xf3
:0203000000080xf3
:020310FFFFFFFF1898021ACCFF0246FFFFFFFF14
/home/pi/matala_energia/Project_Python_source_file/Serial_Sensor.py:154:
Warning: Data truncated for column 'dt' at row 1  cursor.execute(sql_raw)
update moisture_counter set counter='54';
updated!
```

Kuvio 21. Tietokannan virheilmoitus

Huomattiin, että tietokannassa on jotain väärin, koska ohjelma ei pystynyt lähettämään tietoa tietokantaan. Korjattiin tietokannasta mittaus_device structure:sta dt kolumnin, vaihdettiin sen time:stä timestamp:ksi ja defaultiksi CURRENT_TIMESTAMP.

Jonka jälkeen Multible_com.py toimii laboratorio olosuhteissa. **(Kuvio 22.)** Ohjelma tallentaa päivämäärän, ottaa yhteyden tietokantaan. Jonka jälkeen avaa portin ja lähettää kyselyt. Sen jälkeen vastaanottaa tiedot ja lähettää tiedot tietokantaan.

```
python Multible_com.py
I give password Input password
(counter is going to be updated)
previous counter is 73. if successed it is going to be 74
 1, Date stored
 2, Database connected
 3, Managed to function.
 * Serial (/dev/ttyUSB0) can be started
:0203000000080xf3
:0203000000080xf3
update moisture_counter set counter='74';
updated!
```

Kuvio 22. Toimiva koodi.

3.2.3 Tietokannan siirto

Jotta saataisiin molemmat talot samaan tietokantaan, siirrettiin vanhemmat tietokannat uuteen tietokantaan. Koska ei voinut ilman oikeuksia siirtää niitä

suoraan uuteen tietokantaan, vaan olisi maksimissaan pystynyt siirtämään 2000 KB, ladattiin tiedot .sql –muodossa tietokannasta tietokoneelle. Kokoa sille tuli noin 36 MB ja se sisälsi noin 380 000 riviä tietoa. Tehtiin ohjelma **(Kuvio 23.)** joka muokkaisi niitä samalla, kun se siirtää tiedot uuteen tietokantaan.

Ohjelma **(Kuvio 23. ja 24.)** ottaa mittaus_device_10000.txt tiedostosta jokaiselta riviltä kaikki kohdat jotka on erotettu pilkuilla, jonka jälkeen laittaa kyseiset arvot muuttujiin. Joita muokkaamalla eri tavoilla, saadaan haluttu lopputulos niiden arvoihin. Esimerkiksi inner_temp = -40.1 + (0.01 * 6194). Jolloin inner_temp muuttujan arvoksi tulee 21.84.

Sen jälkeen ohjelma avaa yhteyden tietokantaan ja lähettää ne sinne. Ohjelma myös tallentaa kyseiset sql-lausekkeet erilliseen tiedostoon varmuuden vuoksi.

```
import MySQLdb
#Tietokannan hostin nimi, portti, käyttäjä, salasana, tietokanta
db = MySQLdb.connect(host="" , port=, user="", passwd="",db="")
#Avataan sql-tiedosto, mihin tallennetaan sql-lauseet
fwrite = open('testiresult.sql', 'a')
#Avataan tekstitiedosto, josta otetaan tiedot
fread = open('mittaus_device_10000.txt', 'r')
#Luetaan rivi kerrallaan tiedostosta
lines = fread.readlines()
for i in lines:
    #Laitetaan 11 muuttujaan tekstitiedostosta data, data on erotettu pilkulla tekstitiedostossa.
    ttime,addr,counter,inner_temp,inner_hum,outer_temp,outer_hum,outside_temp,outside_hum,device_temp,device_hum = i.split(", ")
    #Alustetaan muuttujat int muotoihin.
    addr = int(addr)
    counter = int(counter)
    inner_temp = int(float(inner_temp))
    inner_hum = int(float(inner_hum))
    outer_temp = int(float(outer_temp))
    outer_hum = int(float(outer_hum))
    outside_temp = int(float(outside_temp))
    outside_hum = int(float(outside_hum))
    device_temp = int(float(device_temp))
    device_hum = int(float(device_hum))
    c1=-4
    c2=0.0405
    c3=-0.0000028
    t1=0.01
    t2=0.00008
    d1=-40.1
    d2=0.01
```

Kuvio 23. SQL-vientiohjelman koodi osa 1

```
#Muokataan muuttujia, että saadaan oikeat lämpötilat ja kosteusprosentit.
inner_temp=d1 + (d2 *inner_temp)
inner_hum=c1 + (c2 *inner_hum) + (c3 *(inner_hum*inner_hum))
inner_hum=(inner_temp-25)*(t1+t2*inner_hum)+inner_hum
outer_temp=d1 + (d2 *outer_temp)
outer_hum=c1 + (c2 *outer_hum) + (c3 *(outer_hum*outer_hum))
outer_hum=(outer_temp-25)*(t1+t2*outer_hum)+outer_hum
outside_temp=d1 + (d2 *outside_temp)
outside_hum=c1 + (c2 *outside_hum) + (c3*(outside_hum*outside_hum))
outside_hum=(outside_temp-25)*(t1+t2*outside_hum)+outside_hum
device_temp=d1 + (d2 *device_temp)
device_hum= c1 + (c2 *device_hum) + (c3 *(device_hum*device_hum))
device_hum=(device_temp-25)*(t1+t2*device_hum)+device_hum
#Kirjoitetaan .sql tiedostoon sql-lausekkeet.
fwrite.write("insert into a_talo_mittaus_real values ('%s',%d,%d, %f,%f, %f,%f, %f,%f, %f,%f)\n"
% (ttime,addr,counter, inner_temp,inner_hum,outer_temp, outer_hum,outside_temp,outside_hum,device_temp,device_hum))
#Avataan yhteys tietokantaan.
cursor = db.cursor()
#Kirjoitetaan tietokantaan tiedot.
cursor.execute("insert into a_talo_mittaus_real values ('%s',%d,%d, %f,%f, %f,%f, %f,%f, %f,%f)\n"
% (ttime,addr,counter, inner_temp,inner_hum,outer_temp, outer_hum,outside_temp,outside_hum,device_temp,device_hum))
#Lähetetään tiedot tietokantaan ja suljetaan yhteys.
db.commit()
cursor.close()
```

Kuvio 24. SQL-vientiohjelman koodi osa 2

Muokkattiin mittaus_device.sql tiedostoa, poistaen INSERT INTO lauseet, sulut ja heittomerkit. Jotta saisi helpommin otettua rivin kerrallaan tiedostosta ja laitettua tiedot muuttujiin. **(Kuvio 25. ja 26.)**

```
INSERT INTO `mittaus_device` (`dt`, `laite`, `count`, `sensor1_temp`, `sensor1_hum`,
`sensor2_temp`, `sensor2_hum`, `sensor3_temp`, `sensor3_hum`, `device_temp`, `device_hum`) VALUES
('2010-09-27 14:52:52', 1, 1, 6194, 1016, 5279, 1521, 65280, 65280, 6311, 860),
('2010-09-27 14:52:52', 2, 1, 6021, 1187, 5070, 2187, 65280, 65280, 5683, 1388),
('2010-09-27 14:52:52', 3, 1, 65280, 65280, 5062, 2750, 5090, 2544, 6470, 1030),
('2010-09-27 14:52:52', 5, 1, 5762, 1062, 6323, 1076, 5548, 1258, 7292, 542),
('2010-09-27 14:52:52', 6, 1, 5918, 1225, 5087, 2185, 5047, 2103, 6258, 979),
('2010-09-27 14:52:52', 7, 1, 4812, 2733, 4872, 2804, 5393, 2018, 6632, 881),
('2010-09-27 14:52:52', 9, 1, 6093, 944, 6252, 1246, 65280, 65280, 7887, 490),
('2010-09-27 14:52:52', 10, 1, 6070, 1266, 5241, 2164, 65280, 65280, 6227, 1030),
('2010-09-27 14:52:52', 11, 1, 4863, 3042, 5626, 1653, 0, 0, 7498, 536),
('2010-09-27 14:52:52', 4, 1, 65280, 65280, 5338, 1976, 4834, 3035, 6549, 1066),
('2010-09-27 14:52:52', 8, 1, 5495, 2603, 5050, 2754, 5140, 2649, 7426, 596),
('2010-09-27 14:52:52', 12, 1, 6170, 1494, 6752, 905, 65280, 65280, 7708, 582),
```

Kuvio 25. Mittaus_device.sql tiedosto.

```
2010-10-28 06:02:01, 5, 2006, 4748, 2475, 6144, 991, 4708, 1844, 6022, 1047
2010-10-28 06:02:01, 6, 2006, 5608, 1392, 4624, 3234, 4656, 3396, 5710, 1461
2010-10-28 06:02:01, 7, 2006, 4478, 3251, 4616, 2910, 5092, 2355, 6234, 1050
2010-10-28 06:02:01, 9, 2006, 4658, 2961, 6088, 1004, 65280, 65280, 6314, 906
2010-10-28 06:02:01, 10, 2006, 5748, 1207, 4608, 3239, 65280, 65280, 5393, 1716
2010-10-28 06:02:01, 11, 2006, 4588, 3093, 5540, 1494, 0, 0, 6855, 641
2010-10-28 06:02:01, 4, 2006, 65280, 65280, 5310, 1795, 4594, 3057, 5602, 1488
2010-10-28 06:02:01, 8, 2006, 4702, 3492, 4674, 3181, 4773, 2824, 6451, 983
2010-10-28 06:02:01, 12, 2006, 5164, 1991, 4713, 3204, 65280, 65280, 5975, 1199
```

Kuvio 26. Muokattu mittaus_device.txt

Koska tiedostoa muokkattiin pelkästään 'find and replace' komennon avulla Notepad++ ohjelmassa, se jätti tiedostoon paljon tyhjiä rivejä, joita ei haluttu käsin ruveta etsimään ja poistamaan 380 000 rivin joukosta. Tästä johtuen ajoimme powershell skriptin **(Kuvio 27.)**, joka poistaa kaikki tyhjät rivit kyseisestä tekstitiedostosta.

```
(gc mittaus_device.txt) | ? {$_.trim() -ne "" } | set-content mittaus_device_clear.txt
```

Kuvio 27. Powershell skripti.

Koska SQL-vientiohjelma kaatoi tietokoneen, kun yritimme viedä 380 000 riviä dataa kerralla tietokantaan, tehtiin toinen ohjelman **(Kuvio 28.)**, joka pilkkoi kyseisen tiedoston pienempiin osiin, sisältäen 10000 riviä per tiedosto.

```
my_file = 'mittaus_device_clear2.txt'
#Määritetään tiedosto.
sorting = True
hold_lines = []
with open(my_file,'r') as text_file:
    for row in text_file:
        hold_lines.append(row)
#Avataan tiedosto, ja lisätään alkio listaan.
outer_count = 1
line_count = 0
while sorting:
#Järjestetään alkiot listassa.
    count = 0
    increment = (outer_count-1) * 10000
    left = len(hold_lines) - increment
    #Lasketaan rivit nollasta 10000
    file_name = "mittaus_device_" + str(outer_count * 10000) + ".txt"
    #Laitetaan uuden tiedoston nimeksi mittaus_device_10000.txt
    #Ja seuraavaan uuteen tiedostoon 20000, jne.
    hold_new_lines = []
    if left < 10000:
        while count < left:
            hold_new_lines.append(hold_lines[line_count])
            count += 1
            line_count += 1
        sorting = False
    else:
        while count < 10000:
            hold_new_lines.append(hold_lines[line_count])
            count += 1
            line_count += 1
    outer_count += 1
    #Seuraavaan tiedostoon aloitetaan 10001 riviltä tallentamaan.
    with open(file_name,'w') as next_file:
        for row in hold_new_lines:
            next_file.write(row)
#Jatketaan tätä niin kauan kunnes loppuu rivit.
```

Kuvio 28. Tekstinjakajaohjelman koodi.

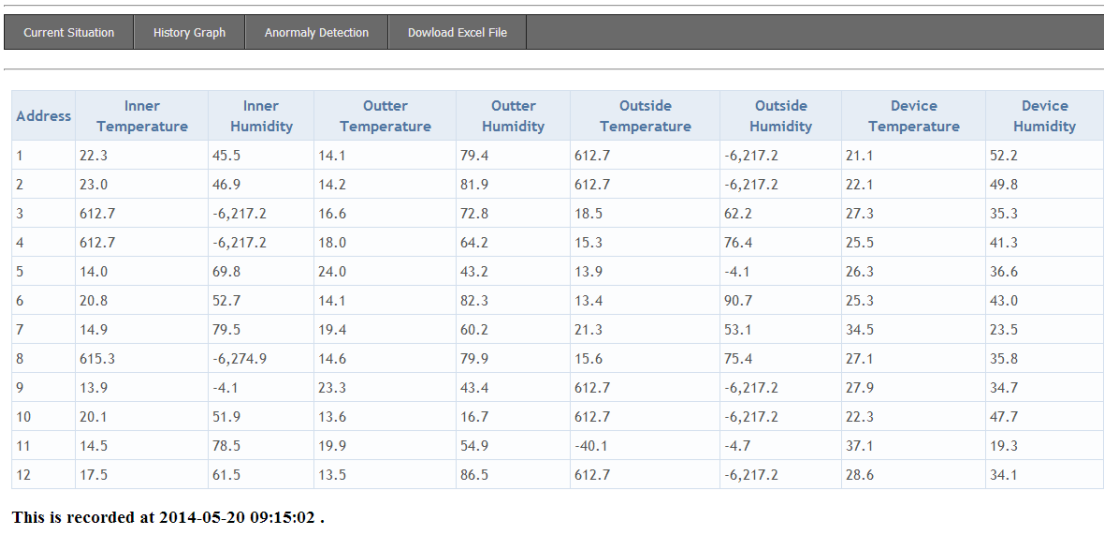
Ohjelma avaa mittaus_device_clear2.txt tiedoston, ja laskee rivit nollasta 10000 asti, jonka jälkeen tallentaa ne rivit mittaus_device_10000.txt tiedostoon. Sen jälkeen se aloittaa uudestaan riviltä 10001 ja tallentaa seuraavat 10000 riviä uuteen tiedostoon nimeltä mittaus_device_20000.txt. Ohjelma jatkaa toimintaansa niin kauan kunnes tiedostosta loppuu rivit.

Ohjelma teki 38 erillistä tekstitiedostoa. Joka tosin teki sen, että SQL-vientiohjelman joutui ajamaan 38 kertaa, että kaikki tiedot sai tietokantaan, mutta tämä oli pikainen ratkaisu kyseiseen ongelmaan.

Tietenkin olisi voinut tehdä sellaisen ohjelman, joka tekee molemmat asiat. Ensimmäiseksi laskee haluttujen rivien määrän, jonka jälkeen lähettää ne muokattuina tietokantaan. Toistaen tätä, kunnes rivit loppuvat tiedostosta.

3.2.5 Internetsivut

Ohjelmille on erikseen PHP:llä Vaasan ammattikorkekoulussa tehdyt internetsivut. **(Kuvio 29.)** Josta näkee tietokantaan viimeksi tallennetut tiedot ja pystyy lataamaan vanhat tiedot Excel tiedostona.



Address	Inner Temperature	Inner Humidity	Outter Temperature	Outter Humidity	Outside Temperature	Outside Humidity	Device Temperature	Device Humidity
1	22.3	45.5	14.1	79.4	612.7	-6,217.2	21.1	52.2
2	23.0	46.9	14.2	81.9	612.7	-6,217.2	22.1	49.8
3	612.7	-6,217.2	16.6	72.8	18.5	62.2	27.3	35.3
4	612.7	-6,217.2	18.0	64.2	15.3	76.4	25.5	41.3
5	14.0	69.8	24.0	43.2	13.9	-4.1	26.3	36.6
6	20.8	52.7	14.1	82.3	13.4	90.7	25.3	43.0
7	14.9	79.5	19.4	60.2	21.3	53.1	34.5	23.5
8	615.3	-6,274.9	14.6	79.9	15.6	75.4	27.1	35.8
9	13.9	-4.1	23.3	43.4	612.7	-6,217.2	27.9	34.7
10	20.1	51.9	13.6	16.7	612.7	-6,217.2	22.3	47.7
11	14.5	78.5	19.9	54.9	-40.1	-4.7	37.1	19.3
12	17.5	61.5	13.5	86.5	612.7	-6,217.2	28.6	34.1

This is recorded at 2014-05-20 09:15:02 .

Kuvio 29. Matalaenergian internetsivut

Vaihdoimme PHP –tiedostoihin vanhojen tietokanta yhteystietojen tilalle uuden tietokannan yhteystiedot, että saadaan uudesta tietokannasta data luettua sivuilta.

4 Testaus

Kytkettyäni Rasperry Pi:n Suvilahden kerrostaloon, testasimme sitä ensimmäiseksi SSH-yhteydellä. Ensimmäiseksi ongelmaksi tuli, kun Multible_com.py -ohjelma ei ottanut yhteyttä mysql.cc.puv.fi -palvelimelle. Kokeilin ping –komennolla löytääkö se ollenkaan kyseistä palvelinta, eikä se löytänyt. Joten muokkasin /etc/hosts tiedostoa komennolla:

sudo nano /etc/hosts

Lisäsimme sinne mysql.cc.puv.fin IP-osoitteen ja nimen, jonka jälkeen ohjelma otti yhteyden. Seuraavaksi ongelmaksi tuli shell.puv.fi -yhteyden luonti, jolloin lisäsimme senkin IP-osoitteen ja nimen /etc/hosts tiedostoon.

Seuraavaksi ongelmaksi tuli sellainen, että virtapiirit eivät saaneet tarpeeksi virtaa. Virtalähde oli vuosien saatossa menettänyt kykynsä tuottaa tarpeeksi virtaa, joten vaihdoimme sen uuteen. Tämän jälkeen virtapiirit saivat sen 24 volttia, jonka ne tarvitsevat toimiakseen.

Vaihdoimme Globals.py ohjelman koodiin osoitteet ja portit **(Kuvio 30.)** vastaamaan niihin mihin RS232-muntimet pystyvät lähettämään ja vastaanottamaan tietoa. Jonka jälkeen vaihdoin Serial_Sensor.py ohjelman kyselemien porttien välimatkan. **(Kuvio 31.)**

```
addreses =[[2,6,10,3,7,11,1,5,9],[4,8,12]]
ports = ['/dev/ttyUSB0','/dev/ttyUSB1']
```

Kuvio 30. Ohjelman portit ja osoitteet

```
def ser_performation(self, time, file, db_start, ssh_connect):
    global  information
    for port in range(0, 2):
        for addr in addreses[port]:
            file.mb_log.write("\n"+time.time_formatted+" (%d) "
    print addr
```

Kuvio 31. Porttien välimatka

Sitten laitteet vastasivat **(Kuvio 32. ja 33.)** Multible_com.py ohjelman kyselyihin ajettaessa. Ohjelma tallentaa päivämäärän ja ottaa yhteyden tietokantaan. Jonka jälkeen avaa portin ja lähettää siihen kyselyt. Sen jälkeen vastaanottaa tiedot ja lopuksi lähettää tiedot tietokantaan.

```
I give password Input password
previous counter is 348. if successed it is going to be 349
 1, Date stored
 2, Database connected
 3, Managed to function.
2
Send message to port: /dev/ttyUSB0
:0203000000080xf3

:0203000000080xf3

:020310189E054017B4070CFF00FF001B1503C71A

6
Send message to port: /dev/ttyUSB0
:0603000000080xef

:0603000000080xef

:0603000000080xef

:06031018AF0543197305D11D8602841D6C02DBE7
```

Kuvio 32. Multible_com.py tulostukset 1.

```
4
Send message to port: /dev/ttyUSB1
:0403000000080xf1

:0403000000080xf1

:0403000000080xf1

:040310FF00FF00172F06AF165F07141B03035CE3

8
Send message to port: /dev/ttyUSB1
:0803000000080xed

:0803000000080xed

:0803000000080xed

:080310FFFFFFFF16F006CA16E107021C86029CD3

12
Send message to port: /dev/ttyUSB1
:0c03000000080xe9

:0c03000000080xe9

:0c03000000080xe9

:0C0310191C04FC1A19041AFF00FF001E31021DEF

update a_talo_moisture_counter set counter='349';
updated!
```

Kuvio 33. Multible_com.py tulostukset 2.

Jotta saisimme kyseisen ohjelman ajettua automaattisesti, eikä manuaalista käynnistystä tarvittaisi, laitoin crontab -ohjelman ajamaan sitä 15 minuutin välein.

sudo crontab –e -komento avaa crontabin muokkausta varten, johon laitoin loppuun seuraavanlaisen tekstin:

*/15 * * * * python /home/pi/matalaenergia/Project_Python_source_files/Multible_com.py

Tämä ajaa kyseisen ohjelman 15 minuutin välein.

Toisessa talossa ongelmaksi kehkeytyi, kun ohjelma ei löytänyt /dev/MXUSB/ polkuja. MXUSB on Moxa Uport 1450 RS232/RS485 -muuntimen portteja, joita käytetään ohjelmassa tässä talossa. Löysin Moxan ajureiden kansion, josta luettuani readme -tiedoston, tajusin, että pitää ensimmäiseksi kokeilla ajaa sudo ./upmknod ohjelman, joka laittaa kyseiset polut esille, jos ne ovat hävinneet. Mutta tämä ei korjannut ongelmaa, se kyllä lisäsi kyseiset portit, mutta niihin ei pystynyt lähettämään tietoa.

Seuraavaksi ajoin install -tiedoston, joka heti ajettuani valitti siitä, että Linuxin kernelit on päivitetty ja kyseiset ajurit eivät toimi tässä versiossa. Sen jälkeen katsoin Linuxin kernelin version uname –r –komennolla ja etsin Moxan kotisivuilta sille kernerille sopivat ajurit. Nämä asennettuani, ohjelma pystyi lähettämään tietoa porteille.

5 Yhteenveto

Työ tarjosi hyvän katsauksen Raspberry Pi:hin, Python -kieleen ja ohjelman debuggaukseen. Vaatimuksena oli korjata koko ohjelma, mutta valitettavasti ajan käydessä vähiin ja Suvilahden piirilevyjen toimimattomuuden takia, sain korjattua vain lämpötila-anturi ohjelman. Technobothnia -laboratoriossakaan ei pystynyt testaamaan muuta kuin lämpötila-antureita, joka vaikeutti kaikkien toimintojen korjaamista.

Eniten aikaa meni korjatessa ongelmia. Jos ei heti tajunnut mistä ongelmasta on kyse, piti internetistä etsiä saman tyyppisiä virheilmoituksia ja tulkita niistä, miten se vaikuttaa kyseenomaiseen ohjelmaan. Tämän jälkeen täytyi paikantaa virhe koodista ja samalla tutkia sen rivin ympäristöä, jos sieltä löytyisi jokin muu virhe, minkä takia koodi ei toimi.

Tämä projekti oli kuitenkin erittäin mielenkiintoinen, pelkästään jo Raspberry Pi:n takia. Kyseisellä laitteella kuitenkin on niin mahdottoman paljon mahdollisuuksia tehdä eri asioita. Kokemusta Pythonista ei hirveästi ennen projektia ollut, mutta kuitenkin jonkun verran, että pääsi alkuun. Loppujen lopuksi jäi ihan hyvä mieli projektista, vaikka ei kaikkia vaatimuksia saanutkaan täyteen.

Lähteet

/1/ Raspberry Pi. Viitattu 9.2.2014. http://www.raspberrypi.org/about/

/2/ Raspbian-käyttöjärjestelmä. Viitattu 9.2.2014. http://raspbian.org

/3/ Richardson, M. & Wallace, S. 2012. Getting Started with Raspberry Pi. O'Reilly Media.

/4/ Python-ohjelmointikieli. Viitattu 10.2.2014. https://upload.wikimedia.org/wikipedia/commons/9/91/Python_Programming.pdf

/5/ Geany. Viitattu 10.2.2014. http://www.geany.org/Main/About

/6/ Debugging in Python. Viitattu 26.5.2014. https://pythonconquerstheuniverse.wordpress.com/2009/09/10/debugging-in-python/

/7/ Mittausjärjestelmän periaatekuva. Viitattu 15.4.2014. http://matalaenergia.puv.fi/index.php?page=mittausjaerjestelm-auml

/8/ ICPCON I-7000 RS232-muunnin. Viitattu 14.4.2014. http://www.icpdas.com/products/Remote_IO/i-7000/pictures/i-7520.jpg

/9/ Raspberry Pi malli B. Viitattu 7.4.2014 http://www.raspberrypi.org/wp-content/uploads/2014/03/raspberry-pi-model-b.jpg

/10/ Raspberry Pi:n konfiguraatiosivu. Viitattu 7.4.2014. https://learn.adafruit.com/adafruits-raspberry-pi-lesson-2-first-time-configuration/overview

/11/ Geany logo. Viitattu 19.4.2014. http://www.technohunk.com/wp-content/uploads/2012/11/geany.png